优化HK

$O(m\sqrt{n})$

```
mt19937 gen(time(0));
struct HopKarp {
    int l, r, flow = 0, lim;
    vector<vector<pii>> g;
    vector<pii> match_from_left;
    vector<int> match_from_right;
    HopKarp(int l, int r)
        : l(l),
        r(r),
        g(l),
        match_from_left(l, { -1, -1 }),
        match_from_right(r, -1),
        dist(l), lim((min(l, r) * 95 + 99) / 100) {}

    void add(int u, int v, int ind) { g[u].push_back({ v, ind }); }
    vector<int> dist;
    void bfs() {
        queue<int> q;
        for(int u=0;u<=l - 1;u++) {
            if (!~match_from_left[u].first)
                q.push(u), dist[u] = 0;
            else
                dist[u] = -1;
        }
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for(auto &v: g[u])
                if (~match_from_right[v.first] && !~dist[match_from
                    dist[match_from_right[v.first]] = dist[u] + 1;
                    q.push(match_from_right[v.first]);
                }
```

```
        }
    }
    bool dfs(int u) {
        checkTime();
        if(flag) return 0;
        for(auto &v:g[u])
            if (!~match_from_right[v.first]) {
                match_from_left[u] = v, match_from_right[v.first] =
                return 1;
            }
        for(auto &v: g[u])
            if (dist[match_from_right[v.first]] == dist[u] + 1 &&
                dfs(match_from_right[v.first])) {
                match_from_left[u] = v, match_from_right[v.first] =
                return 1;
            }
        return 0;
    }
    void get_max_matching() {
        for(int it=1;it<=7;it++) {
            bfs();
            int augment = 0;
            for(int u=0;u<=l - 1;u++) {
                if (!~match_from_left[u].first) {
                    int h = dfs(u);
                    augment += h;
                    flow += h;
                    if (flow >= lim) return;
                }
            }
            if (!augment) return;
        }
    }
```

```
    vector<int> get_edges() {
        vector<int> ans;
        for (int u = 0; u < l; ++u)
            if (match_from_left[u].first != -1)
                ans.emplace_back(match_from_left[u].second);
        return ans;
    }
};

void solve()
{
    int n1, n2, m;
    cin >> n1 >> n2 >> m;
    HopKarp sys(n1 + 1, n2 + 1);
    for(int i=1;i<=m;i++) {
        int u, v;
        cin >> u >> v;
        sys.add(u, v, i);
    }
    sys.get_max_matching();
    vector<int> res = sys.get_edges();
    cout << res.size() << '\n';
    for(auto v: res) {
        cout<< v << '\n';
    }
    return;
}
```

KM最大边权匹配

$O(n^3)$

```
//INF= 1e9+7 若求最小权值匹配，应该为-1e9+7;
//const int N = 607;//最多图的点数
ll w[N][N];//边权
bool visa[N],visb[N];//访问标记，是否在交错树中
int match[N];//右部点匹配的左部点（一个只能匹配一个嘛）
int n,delta;
int la[N],lb[N];//左、右部点的顶标
int p[N];
ll c[N];

void bfs(int x){
        int a,y=0,y2=0;
        for(int i=1;i<=n;++i){
                p[i]=0;c[i]=1e9;
        }
        match[y]=x;
        do{
                a=match[y],delta=1e9,visb[y]=1;
                for(int b=1;b<=n;++b){
                        if(!visb[b]){
                                if(c[b]>la[a]+lb[b]-w[a][b]){
                                        c[b]=la[a]+lb[b]-w[a][b];
                                        p[b]=y;
                                }
                                if(c[b]<delta){
                                        delta=c[b];y2=b;
                                }
                        }
                }
                for(int b=0;b<=n;++b){
                        if(visb[b]){
                                la[match[b]]-=delta;lb[b]+=delta;
```

```
                        }else{
                                c[b]-=delta;
                        }
                }
                y=y2;

        }while(match[y]);
        while(y)match[y]=match[p[y]],y=p[y];
}



int KM(){
        for(int i=1;i<=n;i++){
                match[i]=la[i]=lb[i]=0;
        }
        for(int i=1;i<=n;i++){
                for(int j=1;j<=n;j++){
                        visb[j]=0;
                }
                bfs(i);
        }
        ll res=0;
        for(int i=1;i<=n;i++){
                res+=w[match[i]][i];
        }
        return res;
}

int main(){
        while(~scanf("%d",&n)){
                for(int i=1;i<=n;i++){
                        for(int j=1;j<=n;j++){
```

```
                                w[i][j]=read();
                        }
                }
                printf("%d\n",KM());
        }
        return 0;
}
```

ISAP

$O(V^2E)$

```
int gap[N],dep[N];
void bfs(){
	for(int i=0;i<=pnum;i++){
		dep[i]=gap[i]=0;
	}
	queue<int>q;
	q.push(T);
	dep[T]=1;
	gap[1]++;
	while(!q.empty()){
		int now=q.front();q.pop();
		for(int i=fir[now];i;i=l[i].nex){
			if(l[i].w)continue;
			int to=l[i].to;
			if(dep[to]==0){
				dep[to]=dep[now]+1;
				gap[dep[to]]++;
				q.push(to);
			}
		}
	}
}
int cur[N];
ll addflw(int now,ll w){
	if(now==T)return w;
	ll ans=0;
	for(int &i=cur[now];i;i=l[i].nex){
		if(l[i].w==0)continue;
		int to=l[i].to;
		if(dep[to]+1!=dep[now])continue;
		ll tmp=addflw(to,min(w,l[i].w));
		l[i].w-=tmp;l[i^1].w+=tmp;
```

```
                ans+=tmp;w-=tmp;
                if(w==0)return ans;
        }
        cur[now]=fir[now];
        if(--gap[dep[now]]==0)dep[S]=pnum;
        gap[++dep[now]]++;
        return ans;
}

ll isap(){
        ll ans=0;
        bfs();
        memcpy(cur,fir,sizeof(fir));
        while(dep[S]<pnum){
                ans+=addflw(S,1e18);
        }
        return ans;
}
```

预流推进HLLP

$O(n^2\sqrt{m})$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
const int N=2e6+9;
ll read(){
        ll ans=0,f=1;
        char c=getchar();
        while(c<'0'||c>'9'){
                if(c=='-')f=-1;
                c=getchar();
        }
        while(c>='0'&&c<='9'){
                ans=(ans<<1)+(ans<<3)+c-'0';
                c=getchar();
        }
        return ans*f;
}
int n,m,S,T;
int pnum;
ll pfl[N],dep[N],gap[N<<1];//每个点对应的余流，高度；每个高度有多少个
struct cmp{
    bool operator () (int a,int b) const{
        return dep[a]<dep[b];
    }
};
struct line{
        int to,nex;
        ll w;
}l[N<<1];
int fir[N],cntline=1;
void addline(int fr,int to,ll w){
```

```
        cntline++;
        l[cntline].nex=fir[fr];
        fir[fr]=cntline;
        l[cntline].to=to;
        l[cntline].w=w;
}
void add(int fr,int to,ll w){
    addline(fr,to,w);
    addline(to,fr,0);
}
int inq[N];
void bfs(){
        for(int i=0;i<=pnum;i++){
                dep[i]=1e18;gap[i]=0;
        }
    dep[T]=0;
    queue<int>q;
    q.push(T);
    while(!q.empty()){
        int now=q.front();q.pop();
        inq[now]=0;
        for(int i=fir[now];i;i=l[i].nex){
            int to=l[i].to;
            if(l[i].w)continue;//反向跑
            if(dep[to]>dep[now]+1){
                dep[to]=dep[now]+1;
                if(inq[to]==0){
                    q.push(to);
                    inq[to]=1;
                }
            }
        }
    }
```

```
    return;
}
priority_queue<int,vector<int>,cmp>pq;
void push_(int now){
    for(int i=fir[now];i;i=l[i].nex){
        if(l[i].w==0)continue;
        int to=l[i].to;
        if(dep[to]+1!=dep[now])continue;
        ll tmp=min(l[i].w,pfl[now]);//可以推流
        l[i].w-=tmp;l[i^1].w+=tmp;
        pfl[now]-=tmp;pfl[to]+=tmp;
        if(inq[to]==0&&to!=T&&to!=S){
            pq.push(to);
            inq[to]=1;
        }
        if(pfl[now]==0)break;//已经推完了
    }
}//推流
void relabel(int now){
    dep[now]=1e18;
    for(int i=fir[now];i;i=l[i].nex){
        if(l[i].w==0)continue;
        int to=l[i].to;
        if(dep[now]>dep[to]+1){
            dep[now]=dep[to]+1;
        }
    }
}//把u的高度更改为与u相邻的最低的点的高度加1
ll hlpp(){
    bfs();
    if(dep[S]==1e18)return 0;//s与t不连通
    dep[S]=pnum;
    for(int i=1;i<=pnum;i++)if(dep[i]<1e18)gap[dep[i]]++;//统计各个
```

```
    for(int i=fir[S];i;i=l[i].nex){
        if(l[i].w==0)continue;
                int to=l[i].to;
        ll w=l[i].w;
        pfl[S]-=w;pfl[to]+=w;
        l[i].w-=w;l[i^1].w+=w;
        if(inq[to]==0&&to!=S&&to!=T){
            pq.push(to);
            inq[to]=1;
        }
    }//从s向周围点推流
    while(!pq.empty()){
        int now=pq.top();pq.pop();
        inq[now]=0;
        push_(now);
        if(pfl[now]){//还有余流
            gap[dep[now]]--;
            if(gap[dep[now]]==0){
                for(int i=1;i<=pnum;i++){
                    if(dep[i]>dep[now]&&dep[i]<pnum+1&&i!=S&&i!=T){
                        dep[i]=pnum+1;//标记无法到达
                    }
                }
            }//gap优化
            relabel(now);
            gap[dep[now]]++;
            pq.push(now);
            inq[now]=1;
        }
    }
    return pfl[T];
}
int main(){
```

```
    n=read(),m=read(),S=read(),T=read();
    pnum=n+9;
    for(int i=1;i<=m;i++){
        int fr=read(),to=read();
        ll w=read();
                add(fr,to,w);
        }
    printf("%d",hlpp());
    return 0;
}
```

MCMF

```
ll dis[N],pre[N],fl[N];
int inq[N];
int spfa(){
        for(int i=0;i<=pnum;i++){
                inq[i]=0;
                dis[i]=1e18;
                pre[i]=fl[i]=0;
        }
        fl[S]=1e18;
        dis[S]=0;
        queue<int>q;
        inq[S]=1;
        q.push(S);
        while(!q.empty()){
                int now=q.front();q.pop();
                inq[now]=0;
                for(int i=fir[now];i;i=l[i].nex){
                        int to=l[i].to;
                        if(l[i].w==0)continue;
                        if(dis[to]>dis[now]+l[i].c){
                                dis[to]=dis[now]+l[i].c;
                                pre[to]=i;
                                fl[to]=min(fl[now],l[i].w);
                                if(inq[to]==0){
                                        inq[to]=1;
                                        q.push(to);
                                }
                        }
                }
```

```
        }
        return dis[T]<1e18;
}
ll ansfl,anscos;
void mcmf(){
        while(spfa()){
                int now=T;
                for(int i=pre[now];i;i=pre[now]){
                        l[i].w-=fl[T];
                        l[i^1].w+=fl[T];
                        now=l[i^1].to;
                }
                ansfl+=fl[T];
                anscos+=fl[T]*dis[T];
        }
}
```

倍增LCA

给定一棵有根多叉树，请求出指定两个点直接最近的公共祖先。

```
int n,q,rot;
struct line{
        int to,nex;
}l[N<<1];
int fir[N],cntline;
void addline(int fr,int to){
        cntline++;
        l[cntline].to=to;
        l[cntline].nex=fir[fr];
        fir[fr]=cntline;
}
int lg[N];
int dep[N],f[N][25];
void dfs(int now,int f){
        f[now][0]=f;
        dep[now]=dep[f]+1;
        for(int i=1;i<=lg[dep[now]];i++){
                f[now][i]=f[f[now][i-1]][i-1];
        }
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==f)continue;
                dfs(to,now);
        }
}
int LCA(int a,int b){
        if(dep[a]<dep[b])swap(a,b);
        while(dep[a]>dep[b]){
                a=f[a][lg[dep[a]-dep[b]]-1];
        }
        if(a==b)return a;
        for(int i=lg[dep[a]]-1;i>=0;i--){
```

```
                if(f[a][i]!=f[b][i]){
                        a=f[a][i];b=f[b][i];
                }
        }
        return f[a][0];
}

void solv(){
        n=read(),q=read(),rot=read();
        for(int i=1;i<=n;i++){
                lg[i]=lg[i-1]+(1<<lg[i-1]==i);
        }
        for(int i=1;i<n;i++){
                int fr=read(),to=read();
                addline(fr,to);
                addline(to,fr);
        }
        dfs(rot,0);
        for(int i=1;i<=q;i++){
                int a=read(),b=read();
                printf("%d\n",LCA(a,b));
        }
}
```

tarjan求强连通分量

```
int dfn[N],low[N],dfncnt;
int top,stk[N],instk[N];
int col,bel[N],w[N],wc[N];
void tarjan(int now){
        dfn[now]=low[now]=++dfncnt;
        stk[++top]=now;
        instk[now]=1;
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(!dfn[to]){
                        tarjan(to);
                        low[now]=min(low[now],low[to]);
                }else if(instk[to]){
                        low[now]=min(low[now],dfn[to]);
                }
        }
        if(dfn[now]==low[now]){
                col++;
                while(stk[top]!=now){
                        instk[stk[top]]=0;
                        //强连通分量的权重
                        wc[col]+=w[stk[top]];
                        bel[stk[top--]]=col;
                }
                wc[col]+=w[stk[top]];
                sccw[col]=wc[col];
                instk[now]=0;bel[stk[top--]]=col;
        }
}
```

tarjant LCA

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=5e5+9;
ll read(){
        ll ans=0,f=1;
        char c=getchar();
        while(c<'0'||c>'9'){
                if(c=='-')f=-1;
                c=getchar();
        }
        while(c>='0'&&c<='9'){
                ans=(ans<<1)+(ans<<3)+c-'0';
                c=getchar();
        }
        return ans*f;
}
int n,q,rot;
struct line{
        int to,nex;
}l[N<<1];
int fir[N],cntline;
void addline(int fr,int to){
        cntline++;
        l[cntline].to=to;
        l[cntline].nex=fir[fr];
        fir[fr]=cntline;
}
vector<pair<int,int> >vec[N];
```

```
int ans[N];
int fa[N];
int find(int x){
        if(fa[x]==x)return x;
        return fa[x]=find(fa[x]);
}
void dfs(int now,int f){
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==f)continue;
                dfs(to,now);
                fa[to]=now;
        }
        int r=vec[now].size()-1;
        for(int i=0;i<=r;i++){
                int id=vec[now][i].second;
                int to=vec[now][i].first;
                ans[id]=find(to);
        }
}
void solv(){
        n=read(),q=read(),rot=read();
        for(int i=1;i<=n;i++)fa[i]=i;
        for(int i=1;i<n;i++){
                int fr=read(),to=read();
                addline(fr,to);
                addline(to,fr);
        }
        for(int i=1;i<=q;i++){
                int a=read(),b=read();
                vec[a].push_back({b,i});
                vec[b].push_back({a,i});
        }
```

```
        dfs(rot,0);
        for(int i=1;i<=q;i++){
                printf("%d\n",ans[i]);
        }
}


int main(){
        solv();
        return 0;
}
```

树上启发式合并

询问x的子树内，有多少种颜色，至少有k的节点是该颜色

```
int wson[N],siz[N];
int dfn[N][2],dfncnt;
int node[N];
void dfs_siz(int now,int f){
        siz[now]=1;
        dfn[now][0]=++dfncnt;
        node[dfncnt]=now;
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==f)continue;
                dfs_siz(to,now);
                siz[now]+=siz[to];
                if(siz[wson[now]]<siz[to])wson[now]=to;
        }
        dfn[now][1]=dfncnt;
}
int c[N],a[N];
vector<pair<int,int> >q[N];
int ans[N];
int n,m;
struct BIT{
        int c[N];
        //单点修改
        void init(){
                memset(c,0,sizeof(c));
        }
        void add(int pos,int k){
                if(pos==0)return;
            for (int i = pos;i <= N-1;i += i&-i) c[i] += k;
        }
        //区间查询
        int query(int x){
```

```
            int ans = 0;
            for (int i = x;i;i -= i&-i) ans += c[i];
            return ans;
        }
        int query(int l,int r){
            return query(r)-query(l-1);
        }
}TR;

void dfs_solv(int now,int t,int f){
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==f||to==wson[now])continue;
                dfs_solv(to,0,now);
        }
        if(wson[now])dfs_solv(wson[now],1,now);
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==f||to==wson[now])continue;
                for(int j=dfn[to][0];j<=dfn[to][1];j++){
                        c[a[node[j]]]++;
                        TR.add(c[a[node[j]]],1);
                        TR.add(c[a[node[j]]]-1,-1);
                }
        }
        c[a[now]]++;
        TR.add(c[a[now]],1);
        TR.add(c[a[now]]-1,-1);
        for(int i=0;i<q[now].size();i++){
                ans[q[now][i].second]=TR.query(q[now][i].first,N-1)
        }
        if(t==0){
                for(int j=dfn[now][0];j<=dfn[now][1];j++){
```

```
                        TR.add(c[a[node[j]]],-1);
                        TR.add(c[a[node[j]]]-1,1);
                        c[a[node[j]]]--;
                }
        }
}

void solv(){
        n=read(),m=read();
        for(int i=1;i<=n;i++){
                a[i]=read();
        }
        for(int i=1;i<n;i++){
                int fr=read(),to=read();
                addline(fr,to);
                addline(to,fr);
        }
        for(int i=1;i<=m;i++){
                int ax=read();
                q[ax].push_back({read(),i});
        }
        dfs_siz(1,0);

        dfs_solv(1,0,0);
        for(int i=1;i<=m;i++){
                printf("%d\n",ans[i]);
        }
}
```

给定一颗n个点的带权树，定义一棵树是好的当且仅当这棵树上任意两点间没有一条简单路径满足：路径上的点点权异或和为。

你可以做以下操作若干次：选择一个点，将其的点权更改为任意正整数。请求出最少需要做多少次操作让这棵树变成好的。

```
vector<int>l[N];
int a[N],dis[N];
int n;

int f[N],siz[N],dep[N],wson[N];
void dfssiz(int now,int fa){
        dep[now]=dep[fa]+1;
        siz[now]=1;
        wson[now]=0;
        f[now]=fa;
        dis[now]=dis[fa]^a[now];
        for(int i=0;i<l[now].size();i++){
                int to=l[now][i];
                if(to==fa)continue;
                dfssiz(to,now);
                siz[now]+=siz[to];
                if(siz[wson[now]]<siz[to])wson[now]=to;
        }
}
int ans=0;
set<int>st[N];

void dfs(int now,int fa){
        if(wson[now]){
                dfs(wson[now],now);
                swap(st[now],st[wson[now]]);
        }
        int T=0;
        if(st[now].find(dis[now]^a[now])!=st[now].end())T=1;
        st[now].insert(dis[now]);
        for(auto to:l[now]){
                if(to==wson[now]||to==f[now])continue;
```

```
                dfs(to,now);
                for(auto x:st[to]){
                        if(st[now].find(x^a[now])!=st[now].end()){
                                T=1;
                                break;
                        }
                        for(auto x:st[to]){
                                st[now].insert(x);
                        }
                }
        }
        if(T){
                ans++;st[now].clear();
        }
}

void solv(){
        n=read();
        for(int i=1;i<=n;i++){
                a[i]=read();
                l[i].clear();
                dis[i]=0;
        }
        for(int i=1;i<n;i++){
                int fr=read(),to=read();
                l[fr].push_back(to);
                l[to].push_back(fr);
        }
        dfssiz(1,1);
        ans=0;
        dfs(1,1);
        printf("%d\n",ans);
```

```
}
```

点分治

给定一棵有 n 个点的树，询问树上距离为 k 的点对是否存在。

```
const int ma=1e8+9;
int n,m;
int c[ma];

struct line{
        int to,nex,w;
}l[N<<1];
int fir[N],cntline=1;
void addline(int fr,int to,int w){
        cntline++;
        l[cntline].to=to;
        l[cntline].nex=fir[fr];
        fir[fr]=cntline;
        l[cntline].w=w;
}

int siz[N],wsiz[N];
int vis[N];
int root,Tsiz;
void get_root(int now,int fa){
        siz[now]=1;wsiz[now]=0;
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==fa||vis[to])continue;
                get_root(to,now);
                siz[now]+=siz[to];
                wsiz[now]=max(wsiz[now],siz[to]);
        }
        wsiz[now]=max(wsiz[now],Tsiz-siz[now]);
        if(wsiz[now]<wsiz[root])root=now;
}
vector<int>arr;
```

```
int arrw[ma],arrid[ma];
struct Q{
        int k,w;
}q[N];
void dfs_init(int now,int dis,int fa){
        arr.push_back(dis);
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==fa||vis[to])continue;
                dfs_init(to,dis+l[i].w,now);
        }
}

void cul(int now,int dis,int k){
        arr.clear();
        dfs_init(now,dis,0);
        for(int i=0;i<arr.size();i++){
                if(arr[i]>1e7)continue;
                if(arrid[arr[i]]!=now){
                        arrid[arr[i]]=now;
                        arrw[arr[i]]=1;
                }else arrw[arr[i]]++;
        }
        for(int i=0;i<arr.size();i++){
                for(int j=1;j<=m;j++){
                        int d=q[j].k-arr[i];
                        if(d<0)continue;
                        if(arrid[d]!=now)continue;
                        q[j].w+=arrw[arr[i]]*arrw[d]*k;
                }
        }
}
```

```
void dvd(int now){
        //int nowTsiz=Tsiz;
        cul(now,0,1);
        vis[now]=1;
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(vis[to])continue;
                cul(to,l[i].w,-1);
                //Tsiz=siz[to]>siz[now]?nowTsiz-siz[now]:siz[to];注
                root=0,Tsiz=siz[to];
                get_root(to,0);
                dvd(root);
        }
}

void solv(){
        n=read(),m=read();
        for(int i=1;i<n;i++){
                int fr=read(),to=read(),w=read();
                addline(fr,to,w);
                addline(to,fr,w);
        }
        for(int i=1;i<=m;i++){
                q[i].k=read();
        }

        Tsiz=n; wsiz[0]=1e9; root=0;
        get_root(1,0);
        dvd(root);

        for(int i=1;i<=m;i++){
                if(q[i].w){
                        printf("AYE\n");
```

```
                }else{
                        printf("NAY\n");
                }
        }
}
```

点分树

```
struct line{
        int to,nex;
}l[N<<1];
int fir[N],cntline=1;
void addline(int fr,int to){
        l[++cntline].nex=fir[fr];
        l[cntline].to=to;
        fir[fr]=cntline;
}

struct BIT{
        vector<ll>c;
        int n;
        void init(int _n){
                n=_n;
                c.resize(n+3);
        }
        //单点修改
        void add(int pos,int k){
                //由于dis可能为0,所以在BIT里面统计向后移一位，查询同理
                pos++;//下标[0,n]->[1,n+1]
            for (int i = pos;i <= n;i += i&-i) c[i] += k;
        }
        //区间查询
        ll query(int x){
                x++;//下标[0,n]->[1,n+1]
                x=min(x,n);//注意要和维护的上界取最小值，防止越界
            ll ans = 0;
            for (int i = x;i;i -= i&-i) ans += c[i];
```

```
            return ans;
        }
        ll query(int l,int r){
            return query(r)-query(l-1);
        }
}TR1[N],TR2[N];

int dep[N];
struct LCA{
        int f[N],top[N],wson[N],siz[N];
        void dfs1(int now,int fa){
                siz[now]=1,dep[now]=dep[f[now]=fa]+1;
                for(int i=fir[now];i;i=l[i].nex){
                        int to=l[i].to;
                        if(to==fa)continue;
                        dfs1(to,now),siz[now]+=siz[to];
                        if(siz[wson[now]]<siz[to])wson[now]=to;
                }
        }
        void dfs2(int now,int op){
                top[now]=op;
                if(!wson[now])return;
                dfs2(wson[now],op);
                for(int i=fir[now];i;i=l[i].nex){
                        int to=l[i].to;
                        if(to==f[now]||to==wson[now])continue;
                        dfs2(to,to);
                }
        }
        void init(int root){
                dfs1(root,0);dfs2(root,root);
        }
        int lca(int a,int b){
```

```
                while(top[a]!=top[b]){
                        if(dep[top[a]]<dep[top[b]])swap(a,b);
                        a=f[top[a]];
                }
                if(dep[a]>dep[b])swap(a,b);
                return a;
        }
}T1;

int Dis(int a,int b){
        return dep[a]+dep[b]-(dep[T1.lca(a,b)]<<1);//查询原树中点x,y
}

int siz[N],wsiz[N];
int vis[N];
int root,Tsiz;
void get_root(int now,int fa){//获取该连通块的重心
        siz[now]=1;wsiz[now]=0;
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==fa||vis[to])continue;
                get_root(to,now);
                siz[now]+=siz[to];
                wsiz[now]=max(wsiz[now],siz[to]);
        }
        wsiz[now]=max(wsiz[now],Tsiz-siz[now]);
        if(wsiz[now]<wsiz[root])root=now;
}

int f[N];
void sakura(int now,int fa){//处理重心x所囊括的连通块
        int nowTsiz=Tsiz;
        vis[now]=1;f[now]=fa;
```

```
        //由重心性质可知,TR1会使用[0,now/2],TR2会使用[1,now]，向后移一
        TR1[now].init(Tsiz/2+1);//now到now子树的最大距离
        TR2[now].init(Tsiz+1);//虚树上的fa到now子树的最大距离
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(vis[to])continue;
                Tsiz=siz[to]>siz[now]?nowTsiz-siz[now]:siz[to];//注
                wsiz[root=0]=1e9;
                get_root(to,0);
                sakura(root,now);
        }
}

void change(int x,int val){
        TR1[x].add(0,val);//subtree(x)
        for(int i=x;f[i];i=f[i]){//在虚树上面跳父亲
                int dis=Dis(x,f[i]);
                TR1[f[i]].add(dis,val);
                TR2[i].add(dis,val);
        }
}

ll query(int x,int k){
        ll ans=TR1[x].query(k);//subtree(x)
        for(int i=x;f[i];i=f[i]){//在虚树上面跳父亲
                int dis=Dis(x,f[i]);
                if(dis>k)continue;
                ans+=TR1[f[i]].query(k-dis);
                ans-=TR2[i].query(k-dis);
        }
        return ans;
}
int n,q;
```

```
int a[N];
void solv(){
        n=read(),q=read();
        for(int i=1;i<=n;i++){
                a[i]=read();
        }
        for(int i=1;i<n;i++){
                int fr=read(),to=read();
                addline(fr,to);
                addline(to,fr);
        }

        Tsiz=n;
        wsiz[root=0]=1e9;
        get_root(1,0);
        T1.init(root);
        sakura(root,0);
        for(int i=1;i<=n;i++){
                change(i,a[i]);
        }
        ll preans=0;
        for(int i=1;i<=q;i++){
                ll opt=read(),x=read()^preans,y=read()^preans;
                if(opt){
                        int d=y-a[x];
                        a[x]=y;
                        change(x,d);
                }else{
                        ll ans=query(x,y);
                        preans=ans;
                        printf("%lld\n",ans);
                }
```

```
            }
    }
```

支配树

vectorG[N]就是支配树

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=5e5+10;
ll read(){
        ll ans=0,f=1;
        char c=getchar();
        while(c<'0'||c>'9'){
                if(c=='-')f=-1;
                c=getchar();
        }
        while(c>='0'&&c<='9'){
                ans=(ans<<1)+(ans<<3)+c-'0';
                c=getchar();
        }
        return ans*f;
}
int n,m;
namespace dtree{
        const int N=5e5+20;
        vector<int>E[N],RE[N],rdom[N];
        int S[N],RS[N],cs;
        int par[N],val[N],sdom[N],rp[N],dom[N];

        void clear(int n){
                cs=0;
                for(int i=0;i<=n;i++){
                        par[i]=val[i]=sdom[i]=rp[i]=dom[i]=S[i]=RS[
                        E[i].clear();
                        RE[i].clear();
                        rdom[i].clear();
                }
```

```
}

void addline(int fr,int to){
        E[fr].push_back(to);
}
void Union(int x,int y){
        par[x]=y;
}
int find(int x,int c=0){
        if(par[x]==x){
                return c?-1:x;
        }
        int p=find(par[x],1);
        if(p==-1){
                return c?par[x]:val[x];
        }
        if(sdom[val[x]]>sdom[val[par[x]]]){
                val[x]=val[par[x]];
        }
        par[x]=p;
        return c?p:val[x];
}
void dfs(int x){
        RS[S[x] = ++cs] = x;
        par[cs] = sdom[cs] = val[cs] = cs;
        for(int to :E[x]){
                if(S[to] == 0){
                        dfs(to);
                        rp[S[to]] = S[x];
                }
                RE[S[to]].push_back(S[x]);
        }
}
```

```
int solv(int s,int *up){
        dfs(s);
        for(int i = cs;i;i--){
            for(int e: RE[i]){
                sdom[i]=min(sdom[i],sdom[find(e)]);
            }
            if(i>1){
                rdom[sdom[i]].push_back(i);
            }
            for(int e : rdom[i]){
                int p = find(e);
                if(sdom[p] == i){
                    dom[e] = i;
                }else{
                    dom[e] = p;
                }
            }
            if(i>1){
                Union(i,rp[i]);//这里是i
            }
        }
        for(int i=2;i<=cs;i++){
            if(sdom[i]!=dom[i]){
                dom[i]=dom[dom[i]];
            }
        }
        for(int i=2;i<=cs;i++){
            up[RS[i]]=RS[dom[i]];
        }
        return cs;
    }
```

```
}

int up[N];
vector<int>G[N];
int dep[N],fa[21][N];
void dfs(int x,int f){
        dep[x]=dep[f]+1;
        for(int i=0;i<=19;i++){
                fa[i+1][x]=fa[i][fa[i][x]];
        }
        for(auto to:G[x]){
                if(to==f)continue;
                fa[0][to]=x;
                dfs(to,x);
        }
}
int lca(int x,int y){
        if(dep[x]<dep[y])swap(x,y);
        for(int i=20;i>=0;i--){
                if(dep[fa[i][x]]>=dep[y]){
                        x=fa[i][x];
                }
        }
        if(x==y)return x;
        for(int i=20;i>=0;i--){
                if(fa[i][x]!=fa[i][y]){
                        x=fa[i][x];y=fa[i][y];
                }
        }
        return fa[0][x];
}
```

```
void solv(){
        n=read(),m=read();
        dtree::clear(n);
        for(int i=1;i<=n;i++){
                G[i].clear();
        }
        for(int i=1;i<=m;i++){
                int fr=read(),to=read();
                dtree::E[fr].push_back(to);
        }
        dtree::solv(1,up);
        for(int i=2;i<=n;i++){
                G[up[i]].push_back(i);
        }
        dfs(1,0);
        vector<int>dis(n+1,1e9);
        dis[1]=0;
        queue<int>q;
        q.push(1);
        int T=1;
        while(!q.empty()){
                int now=q.front();q.pop();
                for(auto to :dtree::E[now]){
                        if(dis[to]==1e9){
                                q.push(to);
                                dis[to]=dis[now]+1;
                        }else if(dis[to]!=dis[now]+1){
                                if(lca(to,now)!=to){
                                        T=0;
                                        break;
                                }
                        }
```

```
                }
        }
        if(T){
                printf("Yes\n");
        }else{
                printf("No\n");
        }
}

int main(){
        int tst=read();
        while(tst--){
                solv();
        }
    return 0;
}
```